Published on *Marco de Desarrollo de la Junta de Andalucía* (http://madeja.i-administracion.junta-andalucia.es/servicios/madeja)

Convenio de codificación específico para Java

- **Área:** Especificaciones de Codificación y Construcción
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0015

Se recogen una serie de convenios de codificación específicos de Java, a tener en cuenta además de los generales

En cualquier proyecto de desarrollo es importante emplear una normas de codificación claras y homogéneas para todo el equipo de desarrollo. Con la aplicación de las normas de codificación no se aprecia una mejora funcional, de rendimiento, o de la eficacia, pero son muy necesarias para facilitar la legibilidad del código escrito en lenguaje JAVA, disminuyendo el tiempo a dedicar a las tareas de mantenimiento y corrección.

Pautas

Nomenclatura de clases e interfaces

Usar nombres descriptivos para las clases e interfaces

Los nombres de las clases serán descriptivos y no muy largos, usándose por lo general sustantivos o adjetivos. La primera letra de cada palabra que forme el nombre de la clase deberá estar en mayúsculas, pudiendo introducirse la palabra "Class" al final del nombre de una clase para diferenciarla de las interfaces. Las clases abstractas deben finalizar con la palabra "Abstract". Esta misma normativa deberá emplearse para los nombres de las interfaces.

Volver al índice

Sufijos en la Nomenclatura de Clases e Interfaces

Utilizar los sufijos indicados en función del tipo de clase implementada

Se han definido una serie de sufijos que deberán utilizarse en función del tipo de clase implementada:

- Objeto bean de transporte de datos entre capas (típico POJO): sigue el patrón Value Object, por lo que se define el sufijo **VO**. Ejemplo: DatoVO.

- Objeto bean (POJO) de transporte de datos entre diversas máquinas (por ejemplo, por APIs Web Services). Todos sus elementos implementan Serializable. En este caso el objeto sigue el patrón Data Access Object, sufijo **DTO**. Ejemplo: DatoDTO.
- Objeto de acceso a datos: **DAO** (Data Access Object). Ejemplo: ProvinciaDAO.
- Como regla general, aquellas clases que implementen una funcionalidad característica o un patrón de diseño definido diferente a los anteriormente expuestos, deben indicarlo mediante un sufijo en la denominación del nombre de la clase. El sufijo debe ser lo suficientemente descriptivo para favorecer la compresión de la funcionalidad de la clase. Por ejemplo, si creamos una clase para manejar las operaciones de una cuenta bancaria y empleamos el patrón SessionFacade, habría que denominar a la clase como AccountSessionFacade

En función de una separación entre la interfaz y su implementación:

Interfaces y clases que implementan la interfaz no llevan prefijo o sufijo (no utilizaremos I como prefijo) y las clases que implementan una interfaz llevan el sufijo **Impl**

Volver al índice

Nomenclatura de identificadores

Los identificadores presentes en el código Java deben seguir una normas específicas en su nomenclatura

Se permite el uso del carácter (_) en los identificadores, excepto al principio o al final y nunca se podrán poner dos seguidos.

Volver al índice

Uso de Get y Set

Emplear "get" y "set" para consultar o modificar atributos respectivamente

Se podrán emplear los términos "get" y "set" para identificar los métodos encargados, respectivamente, de consultar o establecer los atributos de la clase.

Volver al índice

Comentarios sobre la implementación

Incluir comentarios en el código indicando las decisiones de implementación

Se deben incluir comentarios en la implementación para facilitar la comprensión del código, facilitando el mantenimiento, la mejora de las aplicaciones y la reutilización del código. Estos comentarios describirán por qué se toman ciertas decisiones en la implementación.

Volver al índice

Estructura de paquetes

Estructurar el código de la aplicación en paquetes según el patrón dado

Los paquetes de las aplicaciones Java desarrolladas para las distintas consejerías deben seguir el siguiente patrón:

```
es.juntadeandalucia.CONSEJERIA.APLICACION.[SUBSISTEMA].[CAPA].[TIPO]
```

evitando que existan clases en el paquete ráiz. Esta nomenclatura permitirá mejorar la comprensión de la estructura de clases de los desarrollos producidos. Habrá que considerar el tamaño de la aplicación para realizar una división por subsistema o directamente dividir por capa la estructura de paquetes de la aplicación.

Quedan fuera de esta norma los paquetes de proyectos reutilizados de otras consejerías.

Volver al índice

Capa en la estructura de paquetes

El patrón CAPA de la estructura de paquetes debe ser presentación, negocio o persistencia

Dentro del patrón para la estructura de paquetes, la CAPA será:

- **presentacion**
- **negocio**
- **persistencia**

Quedan fuera de esta norma los paquetes de proyectos reutilizados de otras consejerías.

Volver al índice

Tipo en la estructura de paquetes

El tipo indicado depende de la capa a la que está asociado

El tipo es variable en función de la capa, así que los paquetes definidos por cada capa son los siguientes:

- **persistencia.dao:** Agrupan las interfaces de los DAO's de la capa de persistencia
- **persistencia.dao.impl:** Implementación de las interfaces de acceso a datos
- **persistencia.entidades:** Agrupa a las clases de entidad que dan origen a las tables en la base de datos
- **persistencia.interfaces:** Agrupa a las interfaces globales (factoría, genérico,...)
- **persistencia.util:** Agrupa a las clases de apoyo (criteria, etc...)
- **negocio.servicios:** Agrupa a las interfaces que separan la lógica de negocio
- **negocio.servicios.Impl:** Agrupa a las clases que implementan las interfaces de lógica de negocio
- **negocio.vo:** Agrupa a la clases encargadas de transporte de datos entre capas
- **negocio.dto:** Agrupa a la clases de transporte de datos entre diversas máquinas
- **negocio.util:** Agrupa a las clases de apoyo (excepciones, autenticación....)
- **presentacion.util:** Utilidades de apoyo a la capa de presentación (validadores personalizados, etc...)
- **presentacion.controlador:** Agrupa a las interfaces de los action que produce JSF
- **presentacion.controlador.Impl:** Agrupa a las clases de que implementan los action provenientes de JSF

Quedan fuera de esta norma los paquetes de proyectos reutilizados de otras consejerías.

Volver al índice

Nomenclatura de parámetros y variables

No comenzar el nombre de las variables con "_" o "$"

Los nombres de las variables y parámetros nunca empezarán con caracteres como "_" o "$".

Volver al índice

Estructura interna de los ficheros

Crear los ficheros java con la estructura interna que se indica

Los ficheros fuente Java se estructurarán de la siguiente forma:

- Comentarios de inicio
- Sentencia Package
- Sentencias Import
- Cuerpo de la clase o interfaz

Cada fichero Java contendrá una sola clase pública o interfaz. En el caso de clases privadas o interfaces asociadas a una clase pública, se pueden colocar en el mismo fichero que la clase pública, siendo ésta la primera clase del fichero.

Volver al índice

Alineación

Establecer 3 caracteres para la alineación

Se establecerá una sangría de **3 caracteres**, siendo espacios en blanco y no el carácter de tabulador.

Volver al índice

Ubicación de llaves “{ }”

Usar las llaves "{}" de la manera que se expone a continuación

La llave de apertura de un bloque de código irá ubicada siempre al final de la primera línea, mientras que la llave de cierre irá sola en una línea, alineada con la misma columna que la primera línea del bloque. Se aplicará un nivel más de sangría a las sentencias que vayan entre las llaves.

Volver al índice

Codificación ISO-8859-1

Usar la codificación ISO-8859-1 sólo cuando usar UTF-8 no sea posible

Se sugiere que los ficheros con extensión ".properties" sean codificados en ISO-8859-1. Esta excepción también es extensible a los ficheros ".apt", que son utilizados para generar el "composite" en Maven.

Volver al índice

Javadoc

Usar Javadoc para generar la documentación de las aplicaciones

Se debe estandarizar el uso de Javadoc para generar la documentación de las aplicaciones ya que resulta de gran utilidad para la comprensión del desarrollo.

A la hora de incluir los comentarios, es preferible el uso de la tercera persona en los comentarios, ya que la documentación suele estar destinada a un público amplio.

Los comentarios se ubican siempre antes de las clases, métodos, interfaces y atributos a describir

Volver al índice

Pautas

Área: Desarrollo » Especificaciones de Codificación y Construcción

Código	Título	Tipo	Carácter
LIBP-0008	Convenio de codificación general	Directriz	Obligatoria

Recursos

Área: Desarrollo » Especificaciones de Codificación y Construcción

Código	Título	Tipo	Carácter
RECU-0734	Implementación de convenios de codificación en Java	Ejemplo	Obligatorio
RECU-0109	Javadoc	Herramienta	Recomendado